A Stitch in Time and Frequency Synchronization Saves Bandwidth

Anh Luong
Carnegie Mellon University
Pittsburgh, PA
anhluong@cmu.edu

Peter Hillyard
Xandem
Salt Lake City, UT
peter@xandem.com

Alemayehu Solomon Abrar
University of Utah
Salt Lake City, UT
aleksol.abrar@utah.edu

Charissa Che
University of Utah
Salt Lake City, UT
charissa.che@utah.edu

Anthony Rowe
Carnegie Mellon University
Pittsburgh, PA
agr@ece.cmu.edu

Thomas Schmid
University of Utah
Salt Lake City, UT
thomas.schmid@utah.edu

Neal Patwari
University of Utah & Xandem
Salt Lake City, UT
npatwari@ece.utah.edu

ABSTRACT

We specify and evaluate a new software-defined clock network architecture, *Stitch*. We use Stitch to derive all subsystem clocks from a single local oscillator (LO) on an embedded platform, and enable efficient radio frequency synchronization (RFS) between two nodes' LOs. RFS uses the complex baseband samples from a low-power low-cost narrowband transceiver to drive the frequency difference between the two devices to less than 3 parts per billion (ppb). Recognizing that the use of a wideband channel to measure clock frequency offset for synchronization purposes is inefficient, we propose to use a separate narrowband radio to provide these measurements. However, existing platforms do not provide the ability to unify the local oscillator across multiple subsystems. We demonstrate Stitch with a reference hardware implementation on a research platform. We show that, with Stitch and RFS, we are able to achieve dramatic efficiency gains in ultra-wideband (UWB) time synchronization and ranging. We demonstrate the same UWB ranging accuracy in state-of-the-art systems but with 59% less utilization of the UWB channel.

CCS CONCEPTS

• **Computer systems organization → Sensor networks**; **Embedded systems**; • **Hardware → Sensor devices and platforms**;

KEYWORDS

syntonization, ultra-wideband, software-defined platform, sensor networks

ACM Reference Format:
Anh Luong, Peter Hillyard, Alemayehu Solomon Abrar, Charissa Che, Anthony Rowe, Thomas Schmid, and Neal Patwari. 2018. A Stitch in Time and Frequency Synchronization Saves Bandwidth. In *Proceedings of IEEE/ACM Information Processing in Sensor Networks (IPSN 2018).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Clock synchronization is a fundamental requirement for efficient operation of a wide variety of wireless networks. Multiple access methods can be made more efficient when a large-scale network of devices is synchronized. Distributed MIMO systems must either have zero carrier frequency offset (CFO) or incur additional channel overhead and complex digital processing to compensate for non-zero CFO [16]. A wide range of time-based localization systems requires time-synchronized infrastructure devices to be deployed across space. Although cables can be used to distribute a shared clock to infrastructure devices, cabling is impractical for mobile devices and often expensive. In addition, GPS-based synchronization has limited availability when operating indoors or when it is jammed. Wireless clock synchronization is often expensive since most existing methods require large bandwidths that monopolize the radio channel.

More efficient frequency synchronization is particularly compelling for ultra-wideband impulse response (UWB-IR)-based ranging. Although UWB-IR enables highly-accurate localization [27], its transmission occupies gigahertz of spectrum, and due to the low number of channels allocated [1], the UWB-IR channel is quite limited in terms of measurement rate. For example, a conventional ad hoc localization scheme using a leading UWB-IR transceiver with eight tags could have an update rate at most 3.5 times per second [9, p. 21]. This update rate is insufficient for the real-time localization of mobile devices in an ad hoc network. E.g., for quadcopters moving at 5 m/s in a GPS-denied environment, a 3.5 Hz update rate locates devices only every 1.4 m of translation, which may be too infrequent for keeping rotorcrafts in a formation. In this paper, we demonstrate a system that achieves the same ranging accuracy as

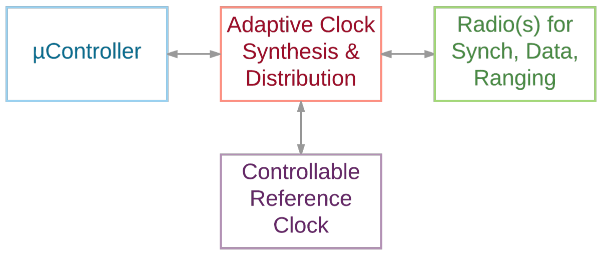
µController
Adaptive Clock Synthesis & Distribution
Radio(s) for Synch, Data, Ranging
Controllable Reference Clock

Figure 1: Stitch is an architecture which generates the clock for each subsystem from a single reference and enables that reference to frequency synchronize with another device using a narrowband radio.

the state-of-the-art UWB-IR system with 59% less utilization of the UWB channel, or equivalently, a 2.4 times higher update rate.

Hidden in plain sight, even low-cost, narrowband transceivers, have the ability to synchronize to the carrier frequency of another device. However, the CFO is rarely exposed, and if so, not with sufficient accuracy, primarily because platforms are not expected to make productive use of that information.

This paper contributes a mechanism for clock to be *syntonized* (frequency synchronized) with other devices using transmissions from a narrowband radio, a method we call *radio frequency synchronization* (RFS). But unless the carrier is generated from the LO, carrier frequency synchronization does not imply LO synchronization. We introduce *Stitch*, a novel platform architecture designed around a software-defined clock network. As depicted in Fig. 1, Stitch adaptively generates the clock for each subsystem from a single controllable reference clock. Combined with RFS and a high-resolution tunable LO, Stitch provides the ability to synchronize two devices' LOs with high accuracy. We evaluate Stitch by developing a research platform and evaluating its use in a time-based ranging system using UWB-IR signals.

In combination, the platforms that use Stitch and RFS can experience dramatic gains in synchronization efficiency, which then allow an increase in the availability of the channel for its primary purpose such as allowing higher data throughput or increasing the number of devices in localization systems.

This paper provides three novel contributions to the synchronization and time-based localization literature across a broad range of wireless networking applications:

(1) We design, implement, and evaluate a new *wireless frequency synchronization system*, called radio frequency synchronization (RFS), which uses low-cost narrowband (NB) transceivers to measure the frequency offset and to synchronize the local oscillators (LOs) on two devices to be within 3 ppb of each other.
(2) We introduce *Stitch*, an architecture that allows clock unification and adaptive distribution across a platform's subsystems. This proposed architecture allows the synchronization of one subsystem to be propagated to the entirety of a platform. Stitch allows the highly syntonized LO provided by RFS to be shared across transceivers.
(3) To demonstrate the advantage of using Stitch and RFS, we implement and evaluate a new protocol, *EffToF*, which minimizes the use of a UWB radio to achieve *bandwidth-efficient wireless time synchronization and time-of-flight (ToF) ranging*. For single-channel single-antenna UWB ranging, we demonstrate ranging RMSE of 17.1 cm, which matches the state-of-the-art [21], while using 59% less of the UWB channel compared to current state of the art.

We implement the three contributions on a research platform that implements the Stitch architecture and provides the ability to adapt the clock distribution network in software. We demonstrate clock unification among multiple subsystems (microprocessors, FPGA, narrowband transceiver, and UWB transceiver). Our platform is a superset of subsystems which could potentially be useful in a range of applications for wireless network synchronization research. The hardware, firmware, and software are open source [3]. Although we use a subset of subsystems on our platform for the UWB evaluation presented in this paper, we anticipate the hardware being useful in other applications.

We first introduce Stitch and RFS in Sections 2 and 3 to show how they enable LO synchronization across subsystems and devices. We motivate, in Section 4, as an example of how using a narrowband radio for frequency synchronization can allow a streamlined UWB ToF algorithm to use 59% less of the UWB channel. We describe a platform for Stitch in Section 5 and use it in Section 6 to quantify its performance.

2 STITCH

In this section, we describe the Stitch architecture and how we use Stitch to enable EffToF. To enable EffToF, we are relying on a secondary narrowband radio to measure the frequency offset of a device's LO compared to another device, and to drive that difference to zero. There are two problems here, one with the clock network on a wireless embedded device, and one with the ability to drive an LO offset to zero. In this section, we describe our solution for the first problem.

When independent oscillators for microcontroller and radio are used, a node has disjoint clock domains. This architecture results in uncertainty (quantization error) in timestamped events especially when an event is generated by one subsystem and timestamped by another [32]. While this is a standard architecture, any synchronization of the radio's LO simply would not benefit the whole node.

One solution would be a single clock source with a bank of frequency synthesizers to fulfill the requirement for a particular combination of radios and microcontrollers contained in the subsystems that are known to be used with the platform. For each combination of ICs and subsystems that could be used, a system designer would need to design an optimal clock tree. This would result in new hardware or a platform tuned around non-standard frequencies impacting the usability of peripheral drivers, etc. Because of that disadvantage, we are motivated to create a platform architecture that allows clock unification and adaptive distribution of a shared clock across a platform's subsystems regardless of what future subsystems may be connected to the main board when it is designed. An example is when we want to attach a daughter-board that requires a 38.4 Mhz clock to a platform with an existing 40 MHz oscillator. If the existing bank of frequency synthesizers on the board cannot provide 38.4 MHz, the two clock domains could not be aligned and hence the processor could not easily timestamp radio events.

We introduce Stitch as a means for future wireless embedded networks to provide highly synchronized clocks across subsystems of a single wireless device and across a network of devices. Current wireless embedded device platforms such as the Raspberry Pi and the Beaglebone have enabled a wide variety of extensible IoT device prototyping, research, and development. However, specialized daughterboards can experience synchronization challenges with current architectures, such as those seen with audio capes for the Beaglebone [2]. Future platforms which use Stitch can enable time and frequency synchronization across a device's subsystems and across a network without prior knowledge of what subsystems will be attached.

The key components of the Stitch architecture are:

(1) **Adaptive clock synthesis**: a field programmable gate array (FPGA) is used for clock synthesis & distribution,
(2) **Controllable reference clock**: a digitally controllable local oscillator which is shared across the entire system, and
(3) **Frequency offset forwarding radio**: a transceiver that allows exportation of either frequency offset estimates or the raw complex baseband samples.

The novelty of Stitch is in the coordination of these known components to achieve the goal of efficient, system-wide and network-wide synchronization. These components work together as follows. Stitch allows quick reconfiguration of hardware through a reprogramming of the low-power low-cost FPGA. A developer can reroute IOs and synthesize the required operating frequencies for individual subsystems from the main reference clock. The FPGA can also be used as a routing table for signal wiring, which increases the adaptability of the platform for other applications.

The controllable reference clock works as an input to the FPGA in order to derive the required clock for each subsystem. Further, any subsystem can tune the reference clock for a particular application requirement.

In combination, these three architectural components allow clock unification across a device's subsystems using commercial off-the-shelf parts. Stitch, as a platform architecture, could be realized in a variety of wireless platforms for a variety of applications. Further, as long as one of the subsystems can tune the LO to match that on another device, clock unification can extend across a wireless network. We describe this wireless clock unification next.

3 RADIO FREQUENCY SYNCHRONIZATION

In this section, we describe a radio frequency synchronization (RFS) mechanism that allows frequency synchronization of two devices' local oscillators (LO) with commercial-off-the-shelf low-power narrowband radios. In short, RFS is so accurate because of: 1) highly accurate CFO estimation performed using the received radio signal; and 2) a unique implementation of a low-cost high-resolution tunable clock source. We describe an implementation using the TI CC1200 in combination with the Beaglebone Black (BBB).

3.1 Frequency Offset Estimation

Several commercially-available low-power radios (e.g., Atmel RF233, Atmel RF215IQ, TI CC1200, Semtech SX1255/7, Silabs EFR32) allow access to either a carrier frequency offset (CFO) measurement or the raw complex baseband (IQ) signal samples, which can be used to estimate the CFO. Since the carriers on the transmitting and receiving devices are generated from the reference clock, this CFO is proportional to the difference between their reference clocks. The application can utilize the CFO estimate to frequency synchronize the receiver's reference clock.

In the case where the IQ signal samples are available, RFS operates on the phase angle of each sample. Assuming that n bits of phase are available, we propose and compare two algorithms to estimate the frequency offset. Both operate on the unwrapped phase integer, which we refer to as p_n, where one cycle corresponds to an integer value between 0 and $2^n - 1$ (which is just a scaled version of the angle in radians). Sample p_n is measured at time $t_n = nT_s$ for sample period T_s, and we expect that it can be expressed as,

$$p_n = \left\lfloor \Phi_{max} \Delta_f t_n + \beta \right\rfloor \mod \Phi_{max}, \tag{1}$$

where $\Phi_{max} = 2^n$, Δ_f is the carrier frequency offset between the transmitter and receiver, and β is the phase offset. Frequency synchronized clocks can be achieved through driving the differences between LOs on the two devices to zero. In Fig. 2, we propose the minimal structure for wireless syntonization with a narrowband transceiver and VCTCXO.

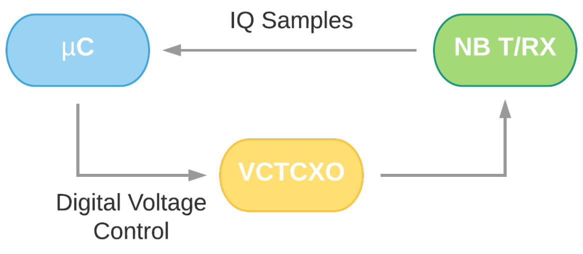

Figure 2: A Beaglebone Black PRU collects complex baseband samples from the narrowband transceiver, computes the frequency offset and corrects the shared VCTCXO. The algorithm iterates until the frequency difference between two nodes is zero.

3.1.1 *Naive Estimator.*

In the "naive" approach, we use the difference in phase over the entire period of N samples; hence, our naive estimation of the carrier frequency is:

$$\Delta_f^n = \frac{p_N - p_1}{\Phi_{max}(N-1)T_s} \tag{2}$$

Due to the quantized nature of the phase measurements, the naive approach has a frequency resolution of $\frac{fs_{iq}}{\Phi_{max}(N-1)}$ Hz, where fs_{iq} is the IQ sampling rate. For example, with $N = 1000, fs_{iq} = 45.044$ kHz, $\Phi_{max} = 2^{10}$, the expected carrier frequency offset resolution is 0.044 Hz.

3.1.2 Linear Regression Estimator. Since the phase noise in p_n fundamentally limits the performance of the estimation of Δ_f, we also implement a low-complexity linear regression method which improves performance with a noisy signal. In summary, we use linear regression to estimate the slope of p_n with n, to which Δ_f is proportional. However, because we do not require the y-intercept and the x-values are regular sampling times, we can significantly reduce the computational requirements and limit the processing to primarily integer multiplications and additions. Generally, a linear regression slope estimate is given by $\frac{\overline{p_n t_n}}{\overline{t_n^2} - (\overline{t_n})^2}$, where $\overline{x_n}$ denotes the sample average of a sequence x_n for $n = 1, \ldots, N$. Since we only care about the slope, any offset of the sampling time axis does not impact the result. Thus we shift the time axis to be zero-mean, in other words, $t_n = T_s\left(n - \frac{N-1}{2}\right)$. With this, $\overline{t_n} = 0$ and $\overline{t_n^2} = \frac{T_s^2 N(N^2-1)}{12}$. Assuming N is odd, the linear regression frequency estimate simplifies to

$$\Delta_f^{lr} = \frac{12}{\Phi_{max} N^2 (N^2-1) T_s} \sum_{n=-(N-1)/2}^{(N-1)/2} n p_n. \tag{3}$$

Calculation of Δ_f^{lr} as given in (3) requires N integer multiplies and adds, and one floating point scale factor.

3.2 LO Frequency Correction

The CFO is a particular fractional multiplication of the reference clock. This constant fraction is written by the application of the transceiver's configuration registers based on the desired carrier frequency, $f_{carrier}$. The PLL multiplier is thus $\frac{f_{carrier}}{f_{LO}}$. With this relationship, the LO offset (xo_{offset}) is directly proportional to the carrier frequency offset (Δ_f) by this PLL multiplier. We simply translate this carrier offset into a corresponding LO offset and increment the VCTCXO control voltage to achieve this LO offset.

Prior to syntonization, the Δ_f may be substantial. Nonlinearities in the DAC and the VCTCXO prevent the algorithm from sending the LO offset to zero in one attempt. Instead, after updating the receiver's LO, the receiver iterates to collect another set of N samples and compute the next LO offset, creating a closed-loop feedback control.

4 EFFICIENT TIME-OF-FLIGHT

In this section, we quantify the benefit of having frequency synchronization performed separately on a narrowband transceiver for the goal of realizing efficient time-of-flight (ToF) ranging and localization systems. We first discuss two existing methods for (ToF) measurement and compare that to efficient time-of-flight (EffToF), our protocol that offloads frequency synchronization and some data communication to a narrowband radio. Using parameters from the Texas Instruments CC1200 narrowband radio and the DecaWave DW1000 UWB radio, EffToF requires 59% less utilization of the UWB channel. For low-latency localization applications, the number of UWB devices is strongly limited due to the high bandwidth of UWB and its relatively long packet duration. Any efficiency gains in the use of the UWB channel allow more devices to be located, or more frequent location measurements.

4.1 Double-sided Two-way Ranging (DS-TWR)

A node (which we call node 1) requires five messages to estimate its offset and skew with respect to the reference node (node 2) through a simple messaging exchange method [15] referred to as the DS-TWR method. As shown in Fig. 3, this method provides each node with two timestamps used for clock offset and skew estimation. Each message carries the previous timestamp; therefore, a "Final" message is required to forward the final timestamp to the node that computes the final solution.

The computation for skew Δ_f and time of flight ToF in the DS-TWR method is:

$$\begin{aligned} \Delta_f &= \frac{T_{RX_2}[i] - T_{RX_2}[i-1]}{T_{TX_1}[i] - T_{TX_1}[i-1]}, \\ ToF &= \frac{T_{RX_1}[i] - T_{TX_1}[i] - T_{TX_2}[i] + T_{RX_2}[i]}{2}, \end{aligned} \tag{4}$$

where $T_{TX_x}[n]$ is time of transmission of message n at node x, and $T_{RX_y}[n]$ is time of reception of message n at node y. The range estimate between node 1 and 2 is simply the ToF multiplied by the speed of light in air; therefore, we refer to both range and ToF estimation interchangeably. Assuming the radio processing delays TX_p and RX_p remain constant, the global time can represented as:

$$T_{RX_2}[i] = T_{TX_1}[i] + TX_p + ToF + RX_p - \epsilon \tag{5}$$

$$T_{RX_1}[i] = T_{TX_2}[i] + TX_p + ToF + RX_p + \epsilon, \tag{6}$$

where ϵ is the clock offset between the two nodes. Here, we still assume the clocks are stable between message exchanges. As a result, the remaining error is primarily limited by the timestamping accuracy.

4.2 PolyPoint Approach

PolyPoint cleverly optimizes packet counts as shown in the middle of Fig. 3 by transmitting a "REF" message immediately followed by a "POLL" message [21]. Since the two messages are identical, node 2 may immediately compute clock skew, which it sends back to node 1. The skew and ToF are given by:

$$\begin{aligned} \Delta_f &= \frac{T_{TX_1}[i] - T_{TX_1}[i-1]}{T_{RX_2}[i] - T_{RX_2}[i-1]} \\ ToF &= \frac{T_{RX_1}[i] - T_{TX_1}[i] - \Delta_f (T_{TX_2}[i] - T_{RX_2}[i])}{2}. \end{aligned} \tag{7}$$

Compared to (4), the protocol in (7) requires floating point multiplication to compute ToF, and as such, requires a more capable processing unit. However, the total number of UWB messages is reduced from five to four, a 20% reduction.

4.3 EffToF

We introduce the efficient time of flight (EffToF) protocol based on the single-sided two-way ranging (SS-TWR) method. The SS-TWR, as suggested in [11, Sec. 12.2], relies on one message exchange which is more time and energy efficient [7]. However, SS-TWR does not provide a means for frequency synchronization. Instead, EffToF leverages the benefit of having frequency synchronized clocks provided by RFS via a secondary narrowband transceiver. EffToF also uses the narrowband radio as a backhaul for UWB packet timestamps. However, a narrowband radio does not produce

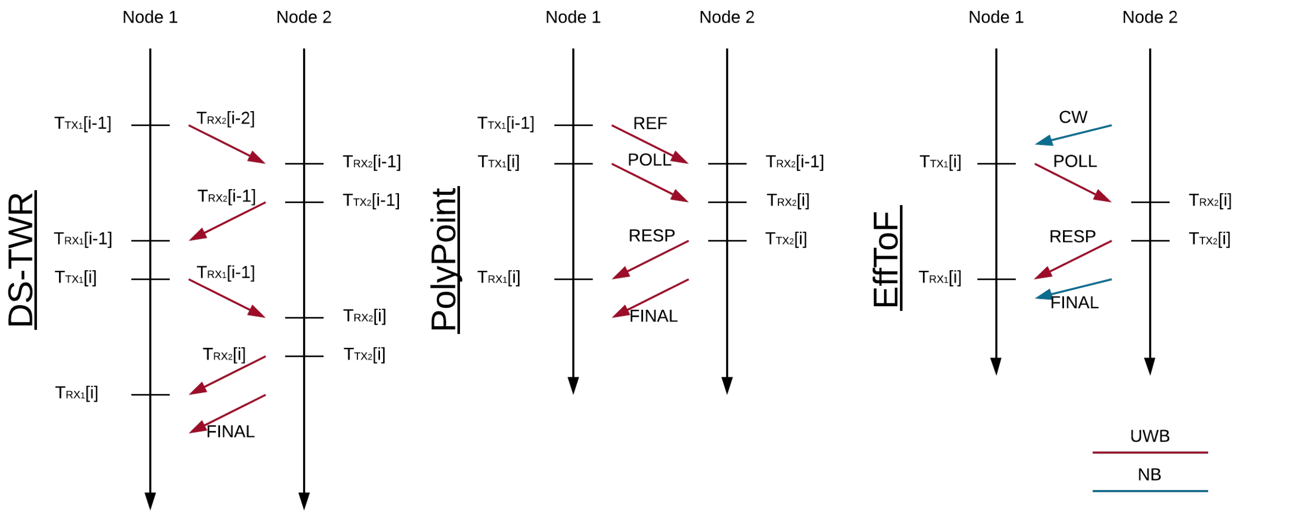

Figure 3: ToF protocols: (Left) conventional message exchange; (Middle) PolyPoint; (Right) proposed EffToF scheme.

high-resolution timestamps; therefore, it can not eliminate the need for messages from the UWB radio in a high-resolution ToF system.

By frequency synchronizing the two clocks, we can reduce the UWB messages to just those required for time of flight estimation. EffToF provides time synchronization and two-way ranging as shown in Fig. 3(right). The red arrows denote transmissions with the UWB transceiver while the blue arrows denote transmissions of the narrowband transceiver. In EffToF, the reference node would transmit a pure tone carrier wave (CW) for devices that wish to frequency synchronize, as shown in Fig. 3(right). Those devices would execute the RFS procedure to correct their frequency offset. Then the SS-TWR operation would allow the node to compute its offset and the ToF. For typical applications and environments, we expect that frequency synchronization would be rare compared to ToF estimation; hence, ToF messaging can be simplified to the 2-way ranging operation. In other words, the POLL and RESP messages are sent with the UWB and the FINAL message is exchanged with the narrowband. The ToF and time offset estimate are given as:

$$ToF \quad = \quad \frac{T_{TX_1}[i] - T_{RX_1}[i] - T_{TX_2}[i] + T_{RX_2}[i]}{2}. \qquad (8)$$

$T_{RX_2}[i]$ and $\delta = T_{TX_2}[i] - T_{RX_2[i]}$ will be sent to the node through narrowband radio message in order to minimize the use of UWB channel.

4.4 Numerical Comparison

For a quantitative comparison, we use the DecaWave DW1000 as our UWB-IR transceiver and the Texas Instruments CC1200 as our narrowband radio. Here we compare the duration of the ToF measurement process for each method, and show that EffToF reduces the UWB channel utilization by 59%.

Each DW1000 UWB's packet contains a long preamble, a long start of frame delimiter (SFD), and slow pulse repetition rate to improve the accuracy of the packet reception's timestamp estimation. DecaWave recommends two settings for ranging [9]:

- **Long-range**: This mode achieves maximum range of as long as 250 m. The data rate is set to 110 kbps with a 1024 symbol-length preamble and a 64-byte SFD.
- **Short duration**: In this mode, the data rate is set to 6.81 Mbps, preamble length to 128 symbols, and SFD to 8 symbols.

With both configurations, the packet duration is calculated as

$$(N_{pre} + N_{SFD})R_{pre} + N_{PHR}R_{PHR} + (8N_d + 48)R_d, \qquad (9)$$

where N_{pre} is the number of symbols in the preamble, N_{pre} is the number of symbols in the start frame delimiter (SFD). R_{pre} is the symbol duration for the preamble (993.59 ns for long-range configuration), N_{PHR} and R_{PHR} are the number (21 symbols) and symbol duration (8205.13ns for the long-range configuration) of the PHR respectively, and N_d and R_d are number of bytes in the message and the bit rate respectively.

In Table 1, we show the content of what should be in the message (function code, node identification and range set number) to help associate timestamps to the round of IEEE 802.15.4 standard exchange. These are the least amount of data for the message exchange in order to compute ToF, time offset (ϵ), and Δ_f. For the sake of simplicity, we have assumed that the target application requires long-range two-way message exchanges. In the following analysis, the DW1000 would be configured to 110 kbps, 1024 preamble symbols, and 64 SFD symbols as the optimal configuration for the long-range operation. In addition to timestamp for ranging, the UWB channel is traditionally used to transfer timestamp data. The final message contains the timestamping information of the last transmission. The preamble and SFD are extraneous for this final message since its timestamp is not utilized for synchronization. The DecaWave DW1000 can be configured for 500 MHz or 1 GHz

System	Msg Type	Bytes	Duration (μs)	Channel	#msg per ToF	Contents
DS-TWR	ALL	21	3026	UWB	5	fn, node#, Ttrx[i], range#
PolyPoint	REF / POLL	13	2501	UWB	2	fn, range#
	RESP	29	3551	UWB	1	fn, node#, Trx[i-1], Trx[i], range#
	FINAL	21	3026	UWB	1	fn, node#, Ttx[i], range#
EffToF	POLL	13	2501	UWB	1	fn, range#
	RESP	14	2566	UWB	1	fn, node#, range#
	FINAL	29	186	NB	1	fn, node#, Trx[i], Ttx[i], range#

Table 1: Message exchange format and duration. Note: DecaWave occupies 500 MHz or 1 GHz bandwidth of an ultra-wide band (UWB) channel, and CC1200 only occupies 12.5 kHz bandwidth of a narrowband (NB) channel.

bandwidth depending on the selection of the channel set. Due to the availability of the channel, the duration of these UWB packets is, in fact, the limiting factor in increasing the number of tags and anchors in the network.

In the example system with n nodes, the ranging rate (r_{range}) can be expressed as $1/(p_{comm}(n-1))$, where p_{comm} is the duration of one round of communication as shown in Fig. 4. Here, we exclude the guard band and discovery slots which further reduce the update rate.

Although this section compares channel utilization for range estimation, we note that generally ToA-based localization systems must compute multiple ranges. If we take the number of ranges to be computed to be A, we can also see that EffToF would similarly make those localization methods more efficient. We compute the channel utilization for a single round of $A = 5$ range measurements in Table 2. Since the CC1200 radio occupies more than 2000x less bandwidth than the UWB transmission, its time-bandwidth product is relatively insignificant and is excluded from Table 2. We only include the minimal duration for message exchanges and exclude the guard time and any additional delays for robust operations. In this particular example, we are able to achieve 69.9% less occupancy of the UWB channel than the DS-TWR approach and 59.2% less than the best-known method for a "Long-range" setup. With "Short duration" configuration, EffTof still provides a 59% and 42% improvement over DS-TWR and Polypoint, respectively.

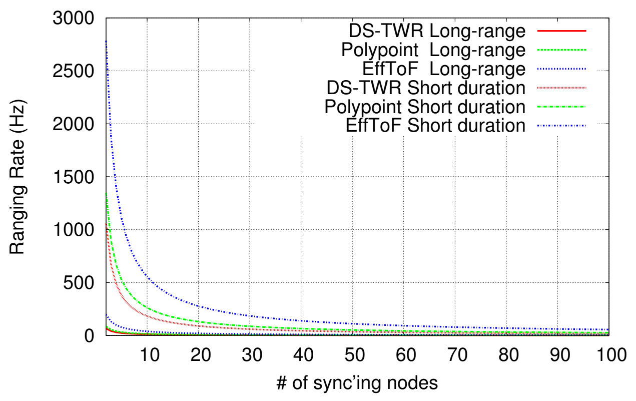

Figure 4: Expected localization update rate versus numbers of synchronized nodes according to Table 2.

5 IMPLEMENTATION

5.1 Hardware

We implemented *PlaStitch*, an ambassador for Stitch, to be an adaptive low-cost low-power platform for sensor network clock-related research as shown in Fig. 5. PlaStitch was designed as a playground for a wide range of clock-related research and development, including the unique ability to change and define the clock network in software, and use multiple radio technologies based on the application without fabricating a complete redesign. If a tested system like EffToF is to be deployed for commercial use, specialized hardware can be built based around the particular subsystems of PlaStitch that are actually required for a target application.

5.1.1 FPGA. The flexibility of PlaStitch comes from a low-power FPGA. We chose to use the Microsemi IGLOO AGL250. We see products such as the Altera SoC, Microsemi SmartFusion, and Cypress PSoC as too large, expensive, and power hungry, for the application requirements of the Stitch architecture. The FPGA is the main mechanism for reconfiguration of the inputs and outputs of clocks and other digital signals between various subsystems. As a result, an FPGA does not need to have an on-chip microprocessor or state-of-the-art performance to implement Stitch. The Microsemi IGLOO optimally meets these requirements. With its 250 MHz 3K logic elements, additional logic could be programmed if required for an application.

5.1.2 Clock Sources. The Microsemi IGLOO FPGA comes with the ability to tune the clock source through its PLL. However, this combination of digital PLL and standard clock source has limited adjustment range. Therefore, PlaStitch uses an external stable 1 ppm 40MHz or 38.4MHz Voltage Controlled Temperature Compensated Crystal Oscillator (VCTCXO) from Abracon, which draws 7.5 mW. We combine the two 12 bit channels of the Microchip MCP4922 to create an overlapping 24 bit, effectively 20 bit, digital-to-analog converter (DAC) to control the Abracon VCTCXO. Over the approximately 800 Hz range of the VCTCXO, this 20-bit control provides a step size of 7.63×10^{-4} Hz. For a 40 MHz LO, this corresponds to 0.02 ppb.

5.1.3 Microprocessors. For raw processing power, PlaStitch offers two main options: 1) Beaglebone-compatible headers, or 2) a dedicated real-time microcontroller. The Beaglebone Black is a popular open-source platform which can run Linux. It provides access to significant pre-existing resources (source code, toolchains,

	Long-range		Short duration	
Protocol	Duration (A, ms)	Duration (A = 5, ms)	Duration (A, ms)	Duration (A = 5, ms)
DS-TWR	$3.026 \times (3A - 1)$	42.364	$0.188 \times (3A - 1)$	2.632
PolyPoint	$2.501 \times 2 + (3.551 + 3.026) \times (A - 1)$	31.310	$0.179 \times 2 + (0.195 + 0.188) \times (A - 1)$	1.89
RFS	$2.501 + 2.566 \times (A - 1)$	12.765	$0.179 + 0.180 \times (A - 1)$	1.085

Table 2: UWB ToF message duration for network of 5 nodes.

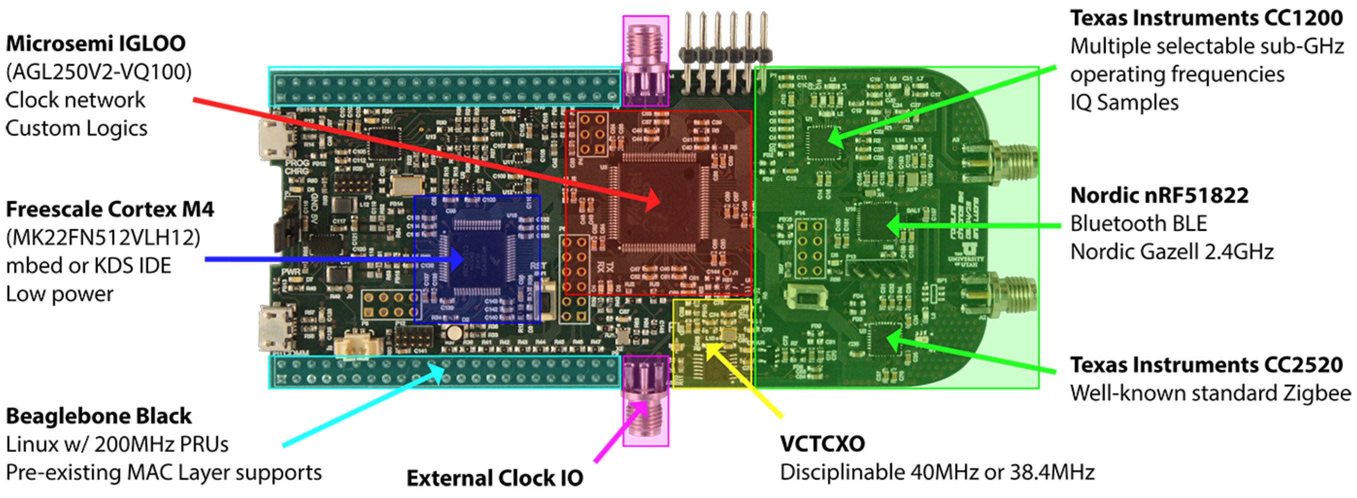

Figure 5: PlaStitch: an adaptable research platform based on Stitch as a Beaglebone cape with a Freescale MK22, Microsemi IGLOO FPGA, multiple radios, 1 ppm 40 or 38.4 MHz VCTCXO, and clock I/O.

communication protocol stacks, etc. . .). PlaStitch also includes a 120 MHz Freescale MK22 microcontroller for low-power real-time applications. The MK22 is a Cortex-M4 with a floating point unit (FPU) and a DSP unit, for that reason, it supports general-purpose signal processing algorithms. The MK22 offers OpenSDA drag-and-drop programming via an on-board Freescale MK20. Moreover, the on-board joint test action group (JTAG) and serial wire debug (SWD) header, and the Beaglebone-compatible SWD connection can also be used as programming interfaces. The Microsemi IGLOO, however, requires an LC Programmer; hence, FlashPro3/4 has to be used with an additional adapter board.

5.1.4 Power. The PlaStitch contains a power management integrated circuit (PMIC) to enable the use of a single cell Li-Po battery for a short-term deployment. Alternatively, it can be powered via micro-USB connection.

5.2 Radio Frequency Synchronization

For RFS, we utilize the Texas Instrument CC1200 on PlaStitch. The TI CC1200 does not directly provide an option to alter the frequency of the local oscillator external to the radio. We rely instead on the *IQ sample* feature of the CC1200. Our recent work has reported on the IQ sample feature [25]; however, we did not use it to estimate the phase. This feature allows the CC1200 to export two registers for the angle of each sample after the CORDIC algorithm. The angle is a 10-bit value, which corresponds to 0.35 degrees resolution.

The sample rate from the CC1200 is limited by the channel bandwidth setting and the maximum speed of the SPI bus. A new IQ sample can only arrive every $T_s = 22.2\mu$s, a sample rate of 45.044 kHz. The data is invalid if the sample is read out while the buffer is being written with another sample. Hence, we check the rising edge of CC1200 `MAGN_VALID` signal, which indicates the availability of the new measurement.

We implement RFS with the Beaglebone Black as the processing unit to demonstrate the feasibility of integrating a Linux-based user application. Using a dedicated embedded microprocessor for RFS is also a possibility; in fact, it is more straightforward to use a real-time processor. The Beaglebone's main processor runs a preemptive Linux OS and thus cannot record the IQ samples from the CC1200 at a precise regular interval. Therefore, we use one of the two real-time co-processors in the programmable real-time processing unit (PRU) sub-system to collect the timing-dependent samples. The PRU supports a very simplified assembly instruction set, which does not allow integer multiplications required for frequency estimation. In this particular case, our user application first configures the radio, then has the PRU store the IQ samples into the shared memory for access by the main processor.

The CC1200 transceiver operates in any of the 169 MHz, 434 MHz, and 900 MHz ISM frequency bands. In our experimental setup, the matching network is populated for 434 MHz, which is our carrier frequency. Our PLL multiplier is thus $\frac{434\ MHz}{f_{LO}}$.

5.3 EffToF

PlaStitch does not have an on-board UWB transceiver to implement EffToF. We exploit the benefit of Stitch rather than completely rebuild the hardware to include an on-board DW1000. We built a DecaWave DW1000 daughter card that operates from an external clock and can be stacked right on top of our platform. We generate a 38.4 MHz to feed both CC1200 and DW1000 through a simple reprogramming of the FPGA and reroute the necessary signals to the microprocessor. Thus, the CC1200 has a PLL multiplication factor of 11.3020833333.

Fig. 6 shows the proposed setup where we use the adaptive clock and digital routing to connect up the subsystems of interest and distribute 38.4 MHz LO to drive both of the subsystems (CC1200 & DW1000).

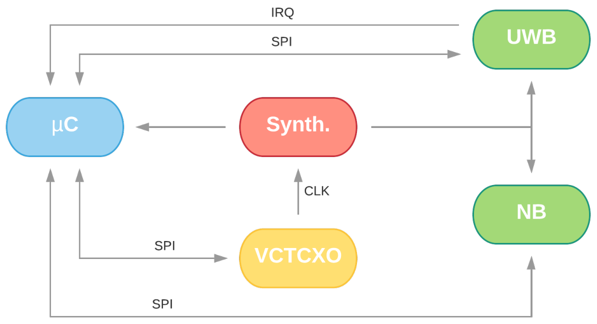

Figure 6: Proposed circuitry and submodules for accurate time synchronization.

6 EVALUATION

In this section, we first present the evaluation of the performance of RFS. Secondly, we quantify the effect of frequency synchronization on ToF measurement. Finally, we evaluate EffToF via four additional experiments.

6.1 Radio Frequency Synchronization

6.1.1 Benchtop. We set a National Instruments vector signal generator (NI VSG) to generate a carrier frequency at 434 MHz. Because the NI VSG is itself not perfect, we must measure its actual carrier frequency. With a Keysight 53230A 12 digits per second frequency counter, we measure the generated frequency to be 434 MHz -62.27 Hz. Not all multiplying factors are possible with a PLL, and with a 40 MHz LO, the closest achievable carrier frequency on the CC1200 is 434 MHz -61 Hz, which occurs when the internal CC1200 PLL has its multiplier and divisor registers set to the have the factor of 10.849998475. We validate this factor by transmitting a pure carrier wave with the CC1200 after setting the previously computed factor and the various local oscillator frequencies. With the aforementioned PLL factor, the target local oscillator frequency should be at 40 MHz -0.117 Hz. Note that the CC1200 has a various intermediate frequency (IF) settings available, we arbitrary select an IF of 138.88 kHz.

As shown in Fig. 7 (left), we approach a frequency difference $<$ 0.1 Hz after the sixth iteration of the naive algorithm (w/ $N = 1000$ samples per iteration) [26].

As seen in Fig. 7 (right), for $N = 1001$, the remaining LO frequency offset is about 0.1 Hz after two cycles of RFS, which for the 40 MHz LO, corresponds to 2.5 ppb. However, we observe worse performance even with the linear regression method, as shown in Fig. 7 (center), due to the noise in the phase measurement.

6.1.2 Long Term Performance. We first conduct a preliminary experiment to quantify the performance of RFS. In this experiment, the nodes are placed on the desk of a clustered office about 1 m apart. The boards are powered on for some time before each experiment to remove any instability during the warm-up period. For this experiment, the two LOs of the two PlaStitch boards remain connected to the frequency counter to keep track of their frequencies.

After just 20 minutes, the free-running uncompensated oscillator skew apart more than 1 Hz (26 ppb) from the other oscillator as shown in Fig. 8. Next, for the same physical setup, we perform RFS once every 20 minutes. We achieve an RMSE of 0.136 Hz. The freerunning method has RMSE of 0.773 Hz. As a result, RFS reduces the clock offset to 3.54 ppb, while freerunning is 20.1 ppb, more than 5× higher, as shown in Fig. 9.

6.1.3 Power and Cost. The power consumption of the Abracon VCTCXO dominates the power consumption budget of the syntonization procedure, as long as syntonization is performed infrequently. The CC1200 uses 46 mA for transmission of a carrier wave at maximum power (14 dBm), which corresponds to 151 mW. The receiver uses half of this power, 75 mW. The MCP4922 DAC and processing unit require 1.5 mW and 2.5 W respectively. Since each syntonization requires 22 ms, one syntonization every 20 minutes requires 45μW. We justify a 20 minute syntonization period with a 24-hour experiment in Section 6. Meanwhile, the Abracon ASVTX-12 requires 7.5 mW.

PlaStitch is designed for research purposes, particularly for extending the hardware lifecycle for time synchronization research by allowing an FPGA-based control of the clock network. The component cost for PlaStitch at the time of publication is about $50. To implement EffToF, however, an FPGA is not required. Since both CC1200 and DW1000 can utilize a 38.4MHz LO, we can reduce cost, complexity, and power by removing the FPGA, which is about $20. Most of the components for RFS are relatively inexpensive compared to the other high stability oscillators we compare in Fig. 13. For quantity one, at the current time, the CC1200, MCP4922, and VCTCXO, cost $6.17, $2.70, and $3.58, respectively. This is $22 (for quantity 1) more than the standard DW1000 tag. For applications with mismatched LO frequencies, a $ 5 frequency synthesizer is a better option than an FPGA, with the exception that it does not allow the clock network to be changed in software. For all of these estimates, high production manufacturing will significantly decrease the costs, particularly for the passives. With that said, for a research platform for applications that are unknown at the time of design, the FPGA is the best option. An example would be a single board IoT platform like the Raspberry Pi to which capes can be attached. The FPGA enables longer hardware usefulness than a single clock or a frequency synthesizer, as new hardware would not need to be designed when a new application or technology must be integrated.

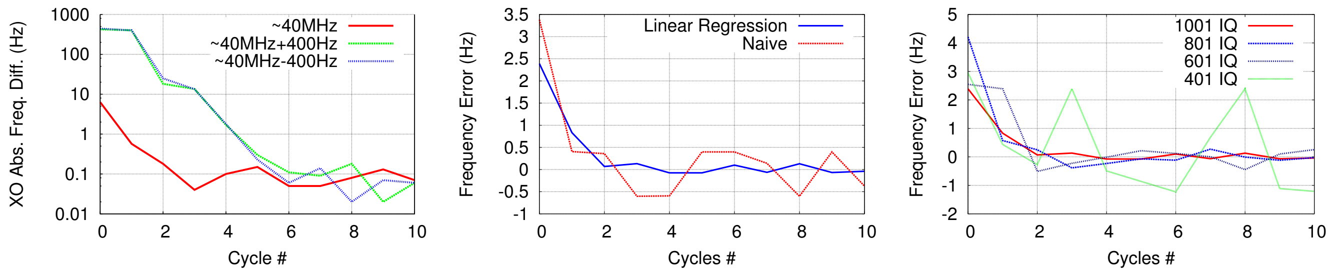

Figure 7: Frequency difference vs. iteration & LO starting frequency (left). The naive algorithm ($N = 1000$) synchs the 40 MHz LO within ± 0.1 Hz in ≤ 6 iterations. Naive vs. linear regression algorithm performace in RFS (center). Frequency synchronization accuracy vs. number of samples N, using the linear regression estimator (right).

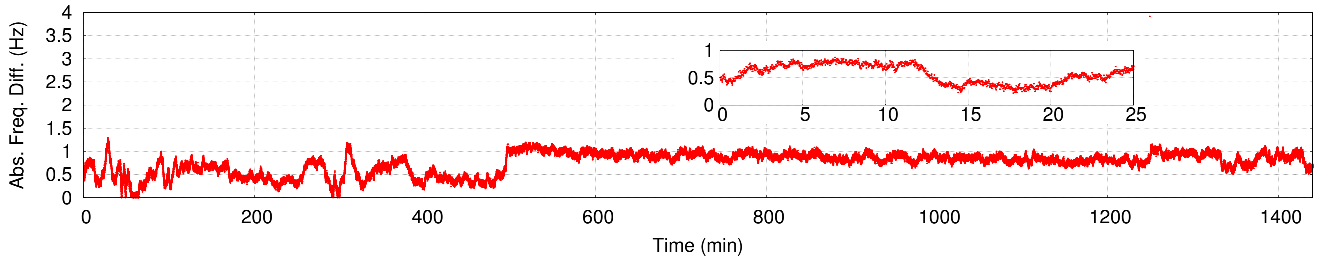

Figure 8: Absolute frequency difference of two uncompensated LOs over a 24 hour period. Typically less than 1 Hz in the first 25 minutes (sub-figure), and after 450 minutes, the frequency offset increases but stays relatively constant.

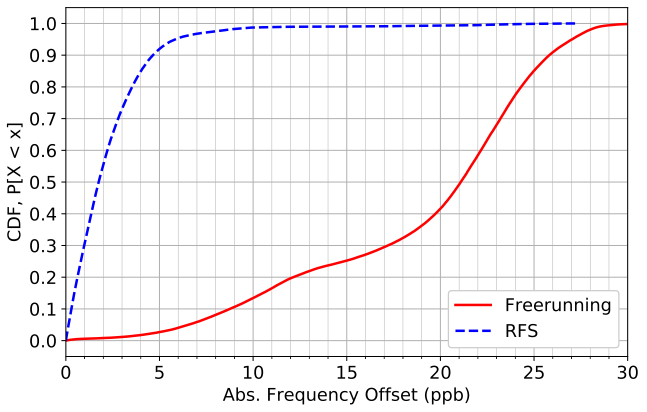

Figure 9: CDF of LO frequency error over 24 hours, freerunning vs. RFS. We achieve an RMSE of 3.54 ppb with RFS running once every 20 minutes while the freerunning method has RMSE of 20.1 ppb.

6.2 Time of Flight

6.2.1 Frequency Offset versus ToF. We build up two DecaWave DW1000 capes, as shown in Fig.11. In the first experiment, instead of using the local oscillator on each of the PlaStitch, we use an arbitrary waveform generator, Rigol DG4102, to generate 38.4 MHz sinusoidal signal on each of the channel, which is connected to each of the cape as the radio reference clock signal. This allows us to generate a known LO frequency offset between two devices. We start each set of measurement by aligning the phase and validating such operation with an oscilloscope. The ToF is measured following the two-way range message exchange as shown in Fig. 3.

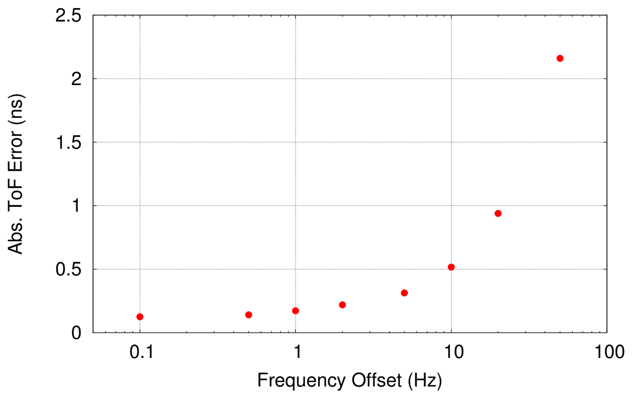

Figure 10: Frequency offset vs. ToF: Absolute ToF error increases with the LO frequency offset.

The frequency of the radio reference clock signal is increased by 0.1 to 50 Hz. At each offset, the ToF between the two UWB devices is measured using SS-TWR. The results are shown in 10. As the frequency offset between the two increases beyond 1 Hz, the ToF (and thus ranging) error increases significantly.

6.3 EffToF Two-way Ranging

We use two identical PlaStitch, Beaglebone Black and DW1000 cape setups as shown in Fig. 11. The UWB antenna, however, is mounted perpendicular to the board rather than parallel to the

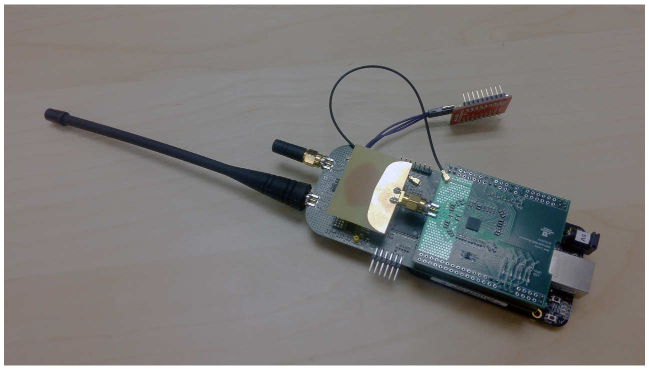

Figure 11: EffToF node: Beaglebone Black (bottom), PlaStitch board (middle), & DW1000 cape (top).

platform as shown in the picture. The devices are calibrated for the correct antenna delay following the recommended procedures [8]. These two nodes are each placed on top of a polymer tool cart, the carts are placed at a distance apart in a large office building. The true distance is measured with a Fluke 414D laser rangefinder with a 50 m maximum range. The ranging error of our meter is less than ±3*mm*. In these experiments, we switch off between the two devices as the reference to remove any device bias. On each node, the FPGA on the PlaStitch synthesizes the LO frequency and routes the required LO signal to both of the CC1200 subsystems and the external DecaWave DW1000 cape. The Beaglebone Black synchronizes the local oscillator with RFS as described in Section 3. The communication for ToF follows the EffToF protocol described in Section 4. A ToF measurement is taken about 2 seconds apart. We record 200 measurements at each true range. When the true range is greater than 50 m, or the line-of-sight (LOS) is blocked, we survey the area and obstructing walls to measure the true distance. The LOS experiments are conducted in different hallways of an office building and inside a partitioned office area. The NLOS tests are conducted through the walls of multiple neighboring offices. Surveying is used to come up with a true distance; hence, cm-level errors will be introduced into the true distance.

Through all of these experiments, we achieve mean absolute error (MAE) of 16.7 cm and root-mean-squared error (RMSE) of 17.1 cm for range measurement shown in Fig. 12. Note, when the actual range is > 35 m, we observe a somewhat higher standard deviation. This may be due to the surveying method used to determine a true distance between the antennas. On top of that, even though the experiments were conducted during hours when few people are present in the building, during the 60 m range experiment, a person walked along the LOS the measurements. This was the only data point with range standard deviation more than 0.13 cm.

SurePoint, an improved version of Polypoint, reports an 8 cm median error in range estimation [20] with frequency and polarization diversity on UWB. In our setup, we have only a single antenna, and we do not hop between channels to diversify measurements. The problem with SurePoint is a dramatically increased number of UWB messages required for each ToF measurement, by a factor of 27. When SurePoint is allowed to use one channel and one antenna, it achieves at best 17 cm median error. EffToF is able to achieve the same level of performance, a 17.5 cm median error

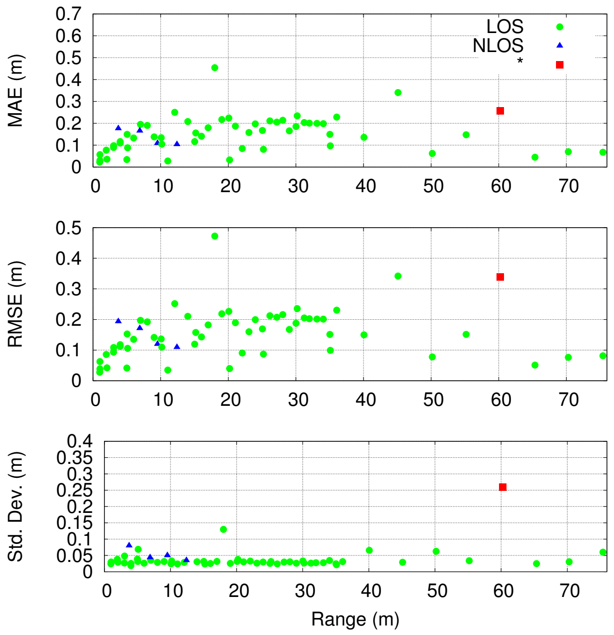

Figure 12: MAE, RMSE, and Standard Deviation of LOS and NLOS ranging experiments. In one experiment (*), a person walked along the link line during the ToF measurement.

with a 3.15 cm standard deviation, across all the ranges in several different environments.

7 RELATED WORK

Message Exchange: There has been a large body of research addressing time synchronization in wireless networks via radio message exchange. The reference broadcast synchronization (RBS) system [13] uses the transmission of a reference message and its reception on the nodes as a marker of epoch for time synchronization. However, the protocol does not consider the propagation delay of the message, which can introduce significant error. The timing sync protocol for sensor networks (TPSN) creates a spanning synchronization tree with nodes that perform pairwise synchronization by two-way message exchange [15]. This then allows nodes to mitigate transmitter and receiver bias. TPSN still assumes that their clock drift during message exchange is negligible. While an additional message exchange would allow a node to estimate its skew, it would increase the utilization of the channel. Flooding-Time Synchronization Protocol (FTSP), Glossy, and PulseSync address the problem of time synchronization for multihop or sizable sparse network with constructive interference through synchronous flooding, thus achieving better synchronization [14, 23, 28]. Fundamentally, clock synchronization still requires a significant amount of messages to converge on an accurate clock's offset and skew.

XO: Wireless devices rely heavily on oscillators for timekeeping (timestamping, tasks scheduling, etc.). A natural approach to achieve more accurate frequency synchronization across devices is to put a highly stable oscillator on each node. Other stand-alone methods to achieve similar accuracy are: GPS-disciplined oscillators [24]; oven controlled oscillators (OCXO) [5]; and chip scale atomic

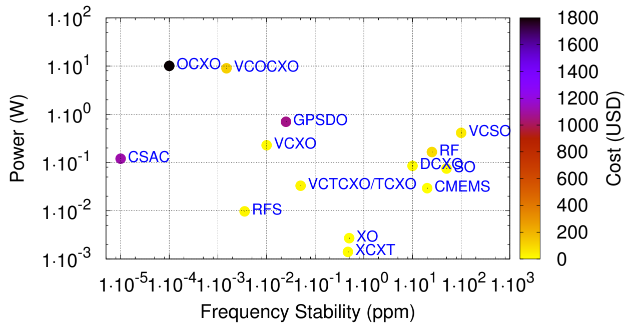

Figure 13: RFS provides a uniquely low-power, low-cost (shown for quantity = 1), and highly stable oscillator, compared to other commercially available oscillators.

clock (CSAC) [29]. All of those are often expensive and have high power consumption [31], as shown in Fig. 13. For instance, the CSAC in [12] uses 120 mW and costs over 1500 USD (quantity of 1). Such specifications preclude their use in many battery-powered and consumer applications. In this paper, we introduce a method for achieving frequency synchronization accuracy on the order of a few parts per billion with low-cost (22 USD quantity of 1) [1] off-the-shelf components and an average power consumption of 7.5 mW.

Reference Reception: A variety of research has addressed frequency synchronization of an oscillator based on a transmitted and received signal. The signal may be ambient or actively transmitted. The ambient electric grid frequency, received by a magnetic field sensor, can train a low-power 32 kHz oscillator [6] which can be used to schedule sleep/wake events. However, such methods have not been used to synchronize the main local oscillators on different nodes. Similarly, an oscillator's frequency can be actively transmitted as a pilot and received by multiple devices [17, 18, 22, 33], which is accurate enough to enable distributed beamforming. Even so, a software defined radio is required for clock synchronization and spectrum regulations may not allow transmission of the desired LO frequency. This is a motivation to transmit over a power line [33] in scenarios where it is convenient to connect each device to a power line. Alternatively, one may transmit the LO at a carrier frequency [34] (or equivalently, transmit two sinusoids separated by the LO frequency [4]). The disadvantage of [4, 34] is that the transmitted signal bandwidth is at least as high as the LO itself. In our case, it would require 38.4 MHz. In comparison, our solution allows the LO to be shared via transmission at a wide range of carrier frequencies, occupies a few tens of Hz of RF bandwidth, and is implemented with standard digital integrated circuits (IC).

Carrier Frequency Synchronization: Carrier frequency synchronization is already an obligatory part of demodulation on most radio receivers [30]. On the TI CC1200, this feature is called *automatic frequency compensation*. When uncompensated, the received complex symbols rotate in the I-Q plane and the symbol error rate increases accordingly. Receivers estimate and correct the phase and frequency offset of the complex baseband signal in a phase-locked loop (PLL). After demodulation, any estimate of the phase or frequency offset is typically discarded. Most transceiver ICs do not provide an option to alter the frequency of the local oscillator external to the radio or export the estimated frequency or phase offsets. In other words, only the radio samples are frequency corrected and only during reception; the rest of the embedded system is unchanged.

Some radios provide a "VCOTUNE" signal to fine tune the external voltage controlled oscillator (VCXO) [10]. Due to the high cost, the on-chip circuitry typically provides a limited tuning range. In case of the DecaWave DW1000, its VCXO tuning range is ±25 ppm with 5 bits resolution, which equals to 1.56 ppm step size. SurePoint [20] is able to limit this tuning range and achieve sub-ppm by using appropriate loading register (23 ppm with 30 steps, 0.79 ppm step size). In order to adjust crystal offset on the fly, they repeat the first packet on each channel, thus, more message exchanges. None of these systems can achieve synchronization on the order of parts per billion because of the high step size. Our solution, on the other hand, enables to tune the VCTCXO with a 0.02 ppb step size and achieve frequency synchronization of less than 3 parts per billion while staying low budget.

Coordinated communication applications: Carrier synchronization is critical in emerging coordinated communication methods such as distributed MIMO [16] and coordinated multipoint (CoMP) [19]. Base stations which use CoMP are recommended [19] to use GPS, when available, or IEEE 1588, which requires expensive hardware and a connection to a high-speed Ethernet network.

MegaMIMO 2.0 is an innovative implementation of distributed MIMO which demonstrates dramatic spectral efficiency gains [16]. MegaMIMO 2.0 computes the carrier frequency offset (CFO) after sampling and counteracts its effects by multiplying by the appropriate complex exponential in real-time digital hardware (Xilinx Zynq Z-7020 FPGA / Cortex A9 processor). The MegaMIMO system introduces the *synchronization trailer*, additional overhead transmitted over the (wideband) channel to help compute the CFO. In comparison, we propose that RFS and Stitch provide, essentially, a common LO across platforms so that CFO does not need to be compensated after sampling. Similar to how EffToF increases the efficiency of UWB ranging, Stitch can enable channel efficiency gains for distributed MIMO and eliminate the need to compensate digitally for CFO, which reduces computational requirements.

8 DISCUSSION

Our system would work in conjunction with other published methods for master selection and network management. In the simplest case, a single device could be chosen to be the master, and the master could rotate to avoid single crystal bias. For a large scale network that is not fully connected, a tree might be constructed to propagate a single frequency through a network. Note that an extension of RFS could transmit an FSK modulated signal rather than a CW signal. In this case, the device's ID could be encoded into the synchronization signal.

RFS, as described here, uses an unmodulated carrier wave for synchronization. Future work could perform synchronization using

[1]We keep a quantity of 1 for a fair comparison with CSAC and OCXO. At a quantity of 1000, the cost is 11 USD.

the IQ samples from a modulated signal, which would allow frequency sync during data transmission, and also potentially allow the receiver to detect and avoid updating the LO when interference power is high.

In addition, other further research could implement RFS on the dedicated logic on the FPGA, which would remove the need to use PRU and the main processing core for this task. In other words, the main processors would be available for other tasks and/or sleep to conserve energy.

In mobile networks, we observe a Doppler shift both while the devices are moving and if there is an object moving in the area of the network. To mitigate this, we would recommend a low syntonization period, for example, syntonizing every few seconds instead of the 20 minute period described in this paper. This allows each LO to remain synchronized despite changes in velocities or motion in the environment.

9 CONCLUSIONS

In this paper, we introduce Stitch, a unique architecture that allows clock unification and adaptive clock distribution across subsystems. We present the PlaStitch, a Stitch-based platform for the testing of a wide range of time- and frequency-synchronized wireless embedded systems. We propose RFS and show that by using low-cost narrowband transceivers, two devices can achieve LO frequency synchronization within 3.5 ppb. With such highly synchronous LOs, it is possible to design ranging protocols that would dramatically reduce the utilization of the UWB channel. We develop and demonstrate the ability to perform efficient time synchronization and two-way time-of-flight ranging, in a protocol we call EffToF. EffToF achieves accurate ranging, with 17.1 cm RMSE at indoor ranges up to 75 m, using 59% less of the UWB channel than the state-of-the-art system. By reducing UWB channel utilization, we enable UWB-based systems to increase the number of time-synchronized devices, or their update rate, or both.

More broadly, this paper contributes to our ability to achieve frequency synchronization on the order of a few parts per billion across a network of devices using low-cost, low-power, and bandwidth efficient methods. We suggest Stitch and RFS be tested in applications such as distributed beamforming and MIMO, improving the robustness of communications infrastructure and navigation systems to GPS jamming, and enabling new highly-synchronous wireless sensing and actuation systems.

10 ACKNOWLEDGMENTS

The authors wish to thank our shepherd Dr. Jie Liu and the anonymous reviewers for providing valuable feedback on this work.

This material is based upon work supported by the US National Science Foundation under Grant Nos. 1329755, 1407949, and 1622741.

REFERENCES

[1] ELECTRONIC CODE OF FEDERAL REGULATIONS. http://www.ecfr.gov (Title 47, Chapter I, Subchapter A, Part 15, Subpart F).
[2] BeagleBone Black with I2S, DSD and SPDIF interface, 2017. http://bbb.ieero.com/.
[3] Plastitch, 2018. https://github.com/SPAN-UofU/Plastitch.
[4] O. Abari, H. Rahul, D. Katabi, and M. Pant. Airshare: Distributed coherent transmission made seamless. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 1742–1750. IEEE, 2015.
[5] Abracon. AOCJY6 Series Datasheet, 2017.
[6] M. Buevich, N. Rajagopal, and A. Rowe. Hardware assisted clock synchronization for real-time sensor networks. In *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*, pages 268–277. IEEE, 2013.
[7] DecaWave. APS007: Wired Sync RTLS With The DW1000, 2017.
[8] DecaWave. APS014: Antennna Delay Calibration, 2017.
[9] DecaWave. DecaRangeRTLS ARM Source Code Guide, 2017.
[10] DecaWave. DW1000 Datasheet, 2017.
[11] DecaWave. DW1000 User Manual, 2017.
[12] A. Dongare and A. Rowe. Propagation-aware time synchronization for indoor applications: Demo abstract. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, pages 292–293. ACM, 2016.
[13] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, Dec. 2002.
[14] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with glossy. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 73–84. IEEE, 2011.
[15] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 138–149. ACM, 2003.
[16] E. Hamed, H. Rahul, M. A. Abdelghany, and D. Katabi. Real-time distributed mimo systems. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*, pages 412–425. ACM, 2016.
[17] D. Jenn, Y. Loke, T. C. Hong Matthew, Y. E. Choon, O. C. Siang, and Y. S. Yam. Distributed phased arrays and wireless beamforming networks. *International Journal of Distributed Sensor Networks*, 5(4):283–302, 2009.
[18] D. Jenn, Y. Loke, M. Tong, E. C. Yeo, and R. Broadston. Distributed phased arrays with wireless beamforming. In *Signals, Systems and Computers, 2007. ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on*, pages 948–952. IEEE, 2007.
[19] V. Jungnickel, K. Manolakis, W. Zirwas, B. Panzner, V. Braun, M. Lossow, M. Sternad, R. Apelfrojd, and T. Svensson. The role of small cells, coordinated multipoint, and massive mimo in 5g. *IEEE Communications Magazine*, 52(5):44–51, 2014.
[20] B. Kempke, P. Pannuto, B. Campbell, and P. Dutta. SurePoint: Exploiting ultra wideband flooding and diversity to provide robust, scalable, high-fidelity indoor localization. In *Proceedings of the 14th ACM Conference on Embedded Networked Sensor Systems*, SenSys'16, November 2016.
[21] B. Kempke, P. Pannuto, and P. Dutta. Polypoint: Guiding indoor quadrotors with ultra-wideband localization. In *Proceedings of the 2nd International Workshop on Hot Topics in Wireless*, pages 16–20. ACM, 2015.
[22] I. Kocaman. *Distributed beamforming in a swarm UAV network.* PhD thesis, Monterey California. Naval Postgraduate School, 2008.
[23] C. Lenzen, P. Sommer, and R. Wattenhofer. Pulsesync: An efficient and scalable clock synchronization protocol. *IEEE/ACM Transactions on Networking (TON)*, 23(3):717–727, 2015.
[24] M. A. Lombardi. The use of gps disciplined oscillators as primary frequency standards for calibration and metrology laboratories. *NCSLI Measure*, 3(3):56–65, 2008.
[25] A. Luong, A. S. Abrar, T. Schmid, and N. Patwari. Rss step size: 1 db is not enough! In *Proceedings of the 3rd Workshop on Hot Topics in Wireless*, pages 17–21. ACM, 2016.
[26] A. Luong, T. Schmid, and N. Patwari. Demo abstract: A platform enabling local oscillator frequency synchronization.
[27] D. Lymberopoulos, R. R. Choudhury, J. Liu, S. Sen, X. Yang, and V. Handziski. Microsoft indoor localization competition: Experiences and lessons learned. *SIGMOBILE Mobile Computation and Communication Review (MC2R)*, October 2014.
[28] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi. The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49. ACM, 2004.
[29] Microsemi. Quantum SA.45s CSAC Datasheet, 2017.
[30] M. Rice. *Digital Communications: a Discrete-Time Approach.* Pearson Prentice Hall, 2009.
[31] T. Schmid. *Time in wireless embedded systems.* PhD thesis, University of California Los Angeles, 2009.
[32] T. Schmid, P. Dutta, and M. B. Srivastava. High-resolution, low-power time synchronization an oxymoron no more. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 151–161. ACM, 2010.
[33] V. Yenamandra and K. Srinivasan. Vidyut: exploiting power line infrastructure for enterprise wireless networks. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 595–606. ACM, 2014.
[34] Y. C. Yong. *Receive channel architecture and transmission system for digital array radar.* PhD thesis, Monterey, California. Naval Postgraduate School, 2005.